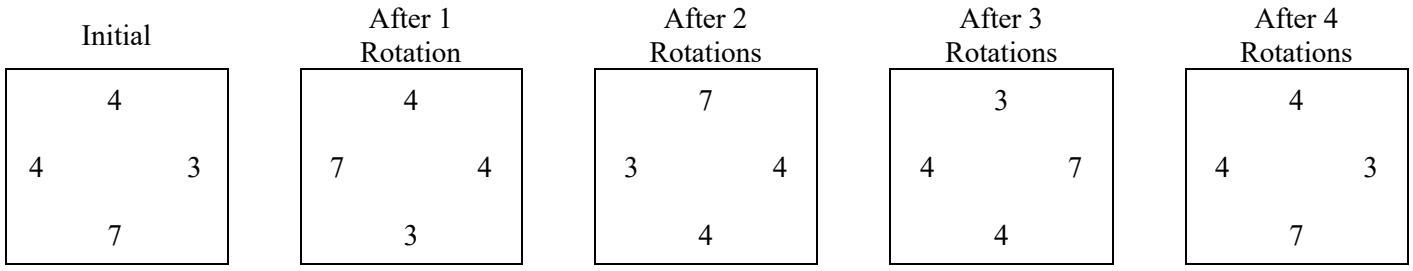
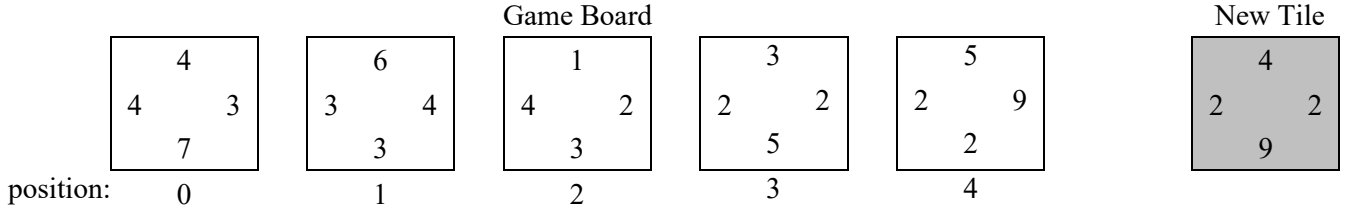
A game uses square tiles that have numbers on their sides. Each tile is labeled with a number on each of its four sides and may be rotated clockwise, as illustrated below.



The tiles are represented by the `NumberTile` class, as given below.

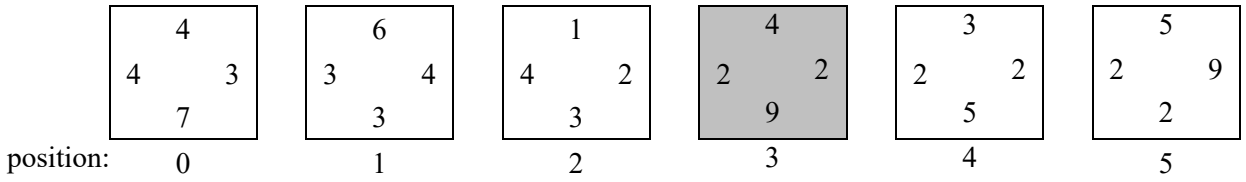
```
public class NumberTile
{
     /**
      * Rotates the tile 90 degrees clockwise
      */
     public void rotate()
     {
          /* implementation not shown */
     }

     /**
      * @return value at left edge of tile
      */
     public int getLeft()
     {
          /* implementation not shown */
     }

     /**
      * @return value at right edge of tile
      */
     public int getRight()
     {
          /* implementation not shown */
     }

       // There may be instance variables, constructors, and methods that are not shown.
}
```

Tiles are placed on a game board so that the adjoining sides of adjacent tiles have the same number. The following figure illustrates an arrangement of tiles and shows a new tile that is to be placed on the game board.



In its original orientation, the new tile can be inserted between the tiles at positions 2 and 3 or between the tiles at positions 3 and 4. If the new tile is rotated once, it can be inserted before the tile at position 0 (the first tile) or after the tile at position 4 (the last tile). Assume that the new tile, in its original orientation, is inserted between the tiles at positions 2 and 3. As a result of the insertion, the tiles at positions 3 and 4 are moved one location to the right, and the new tile is inserted at position 3, as shown below.

GAME BOARD AFTER INSERTING TILE

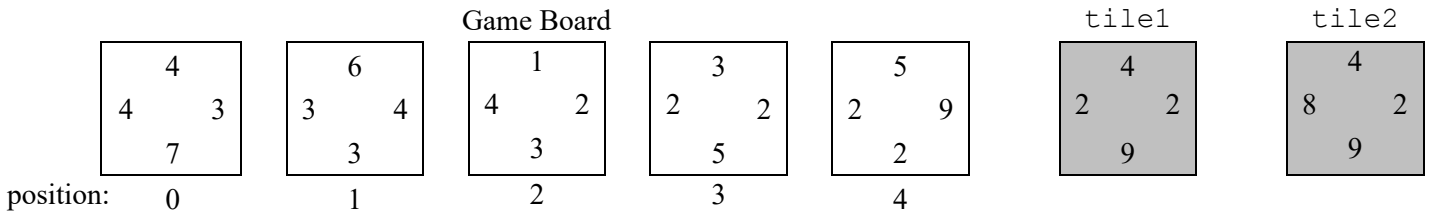


A partial definition of the `TileGame` class is given below.

```
public class TileGame
{
     /** represents the game board; guaranteed never to be null */
     private ArrayList<NumberTile> board;

     public TileGame()
     {
          board = new ArrayList<NumberTile>();
     }

     /** Determines where to insert tile, in its current orientation, into game board
      *  @param tile the tile to be placed on the game board
      *  @return the position of tile where tile is to be inserted;
      *          0 if the board is empty;
      *         -1 if tile does not fit in front, at end, or between any existing tiles;
      *          otherwise, 0 <= position returned <= board.size()
      */
     private int getIndexForFit(NumberTile tile)
     {
          // to be implemented in part (a)
     }

     /** Places tile on the game board if it fits (checking all possible orientations is necessary.
      *  If there are no tiles on the game board, the tile is placed at position 0.
      *  The tile should be placed at most 1 time.
      *  Precondition: board is not null
      *  @param tile the tile to be placed on the game board
      *  @return true if tile is placed successfully; false otherwise
      *  Postcondition: the orientation of the other tiles on the board are not changed
      *  Postcondition: the order of the other tiles on the board relative to each other is not changed.
      */
     public boolean insertTile(NumberTile tile)
     {
          // to be implemented in part (b)
     }

       // There may be instance variables, constructors, and methods that are not shown.
}
```

a) Write the `TileGame` method `getIndexForFit` that determines where a given tile, in its current orientation, fits on the game board. A tile can be inserted at either end of a game board or between two existing tiles if the sides(s) of the new tile match the adjacent side(s) of the tile(s) currently on the game board. If there are no tiles on the game board, the position for the insert is 0. The method returns the position that the new tile will occupy on the game board after it has been inserted. If there are multiple possible positions for the tile, the method will return any one of them. If the tile does not fit anywhere on the game board, the method returns `-1`.

For example, the following diagram shows a game board and two potential tiles to be placed. The call `getIndexForFit(tile1)` can return either 3 or 4 because `tile1` can be inserted between the tiles at positions 2 and 3, or between the tiles at positions 3 and 4. The call `getIndexForFit(tile2)` returns `-1` because `tile2`, in its current orientation, does not fit anywhere on the game board.



Complete the method `getIndexForFit` in the project you downloaded.

---

b) Write the `TileGame` method `insertTile` that attempts to insert the given tile on the game board. The method returns `true` if the tile is inserted successfully and `false` only if the tile cannot be placed on the board in any orientation.

Assume that `getIndexForFit` works as specified, regardless of what you wrote in part (a).

Complete the method `insertTile` in the project you downloaded.